Class06: R Functions

Sam Fisher (A18131929)

Table of contents

Background

Functions are the heart of using R. Everything we do involves calling and using functions (from data input, analysis, to results output).

All functions have at least 3 things:

1. A **name** the thing we used to call the function.
2. One or more input **arguments** that are coma separated.
3. The **body**, lines of code between curly brackets {} that does the work of the function.

A First Function

Let's write a silly wee function to add some numbers:

```
add <- function(x) {
  x + 1
}
```

Let's try it out

```
add(100)
```

```
[1] 101
```

Will this work

```
add(c(100, 200, 300))
```

```
[1] 101 201 301
```

Modify to be more useful

```
add <- function(x, y=1) {
  x + y
}
```

```
add(100, 10)
```

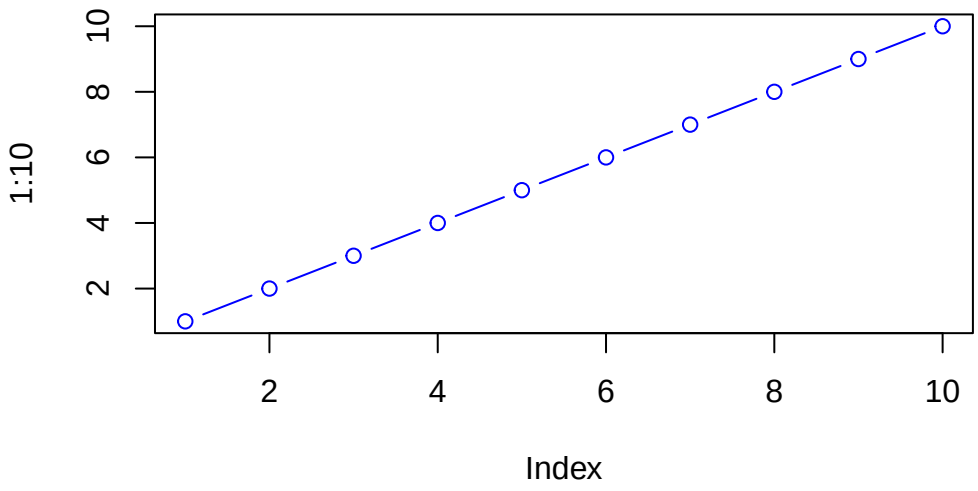
```
[1] 110
```

Will this still work?

```
add(100)
```

```
[1] 101
```

```
plot(1:10, col="blue", typ="b")
```




```
log(10, base=10)
```

```
[1] 1
```

> **Note** Input arguments can either be **required** or **optional**. The later have a fall-back default that is specified in the function code with an equals sign.

A Second Function

All functions in R look like this

```
name <- function(arg) {body}
```

The `sample()` function in R...

```
sample(1:10, 4)
```

```
[1] 9 2 8 6
```

> Q. Return 12 numbers picked randomly from the input 1:10

```
sample(1:10, 12, replace=T)
```

```
 [1]  2 10  2  5  2  6  7  5 10  7  6  2
```

Q. Write the code to generate a 12 nucleotide long DNA sequence?

```
sample(c("A","T", "G", "C"), 12, replace=T)
```

```
 [1] "G" "T" "T" "G" "C" "T" "C" "T" "C" "G" "G" "T"
```

Q. Write a first version function called `generate_dna()` that generates a user specified length `n` random DNA sequence?

```
generate_DNA <- function(n = 10) {
  sample(c("A","T","G","C"), size = n, replace = TRUE)
}
```

```
generate_DNA()
```

```
 [1] "A" "A" "A" "T" "G" "A" "T" "T" "C" "A"
```

Q. Modify your function to return FASTA like sequence so rather than [1] "G" "A" "C" "A" "T" we want "GACAT". We want an actual sequence returned.

```
generate_DNA <- function(n=10) {
  seq <- sample(c("A", "T", "C", "G"), size = n, replace = T)
  paste(seq, collapse = "")
}
```

```
generate_DNA()
```

```
[1] "CACGATATGA"
```

Q. Give the user an option to return FASTA format output sequence or standard multi-element vector format?

```
generate_DNA <- function(n=10, fasta=TRUE) {
  ans <- sample(c("A", "T", "C", "G"), size = n, replace = T)

  if(fasta) {
    ans <- paste(ans, collapse = "")
    cat("Hello...")
  } else {
    cat("... is it me you're looking for...")
  }

  return(ans)
}
```

```
generate_DNA(10)
```

```
Hello...

[1] "CTATCTCTGC"
```

```
generate_DNA(10, fasta=FALSE)
```

```
... is it me you're looking for...

 [1] "C" "T" "C" "T" "A" "A" "G" "A" "C" "A"
```

A new cool function

> Q. Write a function called `generate_protein()` that generates a user specified length protein sequence in FASTA like format?

```
generate_protein <- function(n = 10, fasta = TRUE) {
  amino <- c("A","R","N","D","C","E","Q","G","H","I","L","K","M","F","P","S","T","W","Y","V")
  seq <- sample(amino, size = n, replace = TRUE)

  if (fasta) {
    return(paste(seq, collapse = ""))
  } else {
    return(seq)
  }
}
```

```
generate_protein(15)
```

```
[1] "MGILRDKGCAGMFFE"
```

```
generate_protein(15, fasta=FALSE)
```

```
 [1] "F" "A" "M" "R" "L" "G" "Q" "K" "E" "T" "I" "M" "D" "K" "F"
```

Q. Use your new `generate_protein()` function to generate all sequences between length 6 and 12 amino-acids in length and check if any of these are unique in nature (i.e. found in the NR database at NCBI)?

```
for(i in 6:12) {
  cat(">",i, sep="", "\n")
  cat(generate_protein(i), "\n")
}
```

```
>6
SQMAAM
>7
RRLRPQR
>8
AHVQISWP
>9
YRGSEKSPG
>10
SPWKFRQCRI
>11
WEHQIGKQDTG
>12
LCHDFMCESCGP
```

Identical matches were only seen in the amino acids that were generated with lengths 6 and 7.